SCHREIBEN VERSTÄNDLICHER PROGRAMME

☞ *Ziel:* + *gedankliche Klarheit des Programms offenlegen (--> Fehlervermeidung),*

+ *schnelle Einarbeitung eines Außenstehenden ermöglichen*

☞ *selbstdokumentierende Programme*
+ Vermeiden von Programmiertricks

- ❑ Modularisieren
- ❑ klare Programmgliederung
- ❑ prägnante Kommentare
- ❑ mnemotechnische Bezeichner
- ❑ einheitliche Ausdrucksformen
- ❑ Vermeiden anonymer Direktwerte

1. MODULARISIEREN

☞ *Abgrenzen von Teilfunktionen in separate Programmbausteine (Moduln);*

❑ jeder Modul hat folgende Eigenschaften:

- die Größe ist überschaubar;
- er verwirklicht genau eine Funktion;
- minimale, leicht überschaubare Datenschnittstelle nach außen (Import/Export);
- trifft keine Annahmen über die Wirkungsweise der aufrufenden Umgebung;
- ist verwendbar ohne Kenntnis seiner inneren Wirkungsweise;

❑ jeder Modul hat seine Schnittstellenbeschreibung

- Leistung (Hauptzweck, Spezialfälle seiner Anwendung, Fehlerbehandlung);
- E/A-Parameter;

2. Klare Programmgliederung

- ❑ zeitliches Nacheinander der Abarbeitung drückt sich auch im räumlichen Nacheinander der Programmzeilen aus, d.h. man kann den Programmtext von oben nach unten lesen;
- ❑ Rücksprünge kommen nur zum Zweck der Schleifenbildung vor (➔ (streng-) strukturierte Programmierung);
- ❑ Vorwärtssprünge sind u.U. bei Abbruchbedingungen zweckmäßig (strukturierte Sprünge);
- ❑ Verschachtelung von Anweisungen wird durch Einrücken hervorgehoben!

3. Prägnante Kommentare

☞ *zusätzliche Inhalte, die der Compiler nicht benötigt, aber evtl. ein menschlicher Leser;*

- ❑ Kommentarkopf
  - Bearbeiter, Datum, Versionsnummer
  - Hintergrundinformationen zur Problemlösung, Annahmen über Eingangsdaten;
  - Quellenhinweise zur algorithmischen Lösung
- ❑ Beschreibung wichtiger Datenstrukturen
- ❑ Beschreibung wesentlicher algorithmischer Teilschritte

4. Mnemotechnische Bezeichner

☞ *Mnemonik: Lehre von den Eselsbrücken*

- ❑ vom Bezeichner muß auf Bedeutung des Objekts geschlossen werden können;
- ❑ Programmzeilen mit selbsterklärenden Bezeichnern erübrigen weitere Kommentare;
- ❑ zumutbare Länge (> 6) ?
  ➔ Schreibfaulheit <-> Informationsstau beim Programmieren
- ❑ je größer der Gültigkeitsbereich, um so individueller und damit i.allg. auch länger sollte ein Bezeichner sein;
- ❑ nie mehrere Bedeutungen für einen Bezeichner;
- ❑ keine ähnlichen, leicht zu verwechselnden Bezeichner:
  OTTO / OTT0, ANNA / ANA
- ❑ keine schwer auszusprechenden Bezeichner;

5. Einheitliche Ausdrucksformen

- ❑ identische Namen für gleiche Bedeutungen in allen Moduln/Programmen;
- ❑ gleichartiger Aufbau aller Moduln/Programme;
- ❑ sichtbare Übereinstimmung in vergleichbaren Programmwendungen;
- ❑ in alternativen Zweigen gleiche Reihenfolge analoger Schritte;
- ❑ systematische Vergabe der Bezeichner, so daß aus der Art des Bezeichners auf die Art des Objekts geschlossen werden kann;

6. Vermeiden anonymer Direktwerte

☞ *fördert Änderbarkeit und Lesbarkeit*

- Direktwerte, denen eine eigenständige Bedeutung zugeordnet werden kann, sollten aus dem Anweisungsteil in Form einer Konstantendeklaration herausgezogen werden;

- der Einsatz von Direktwerten sollte auf echte Rechenoperationen beschränkt werden, z.B. i:= i+ 1;